Extending the Capacity of $1/f$ Noise Generation

Guillaume Perez(✉), Brendan Rappazzo, and Carla Gomes

Department of Computer Science, Cornell University, Ithaca, NY 14850, USA
guillaume.perez06@gmail.com, bhr54@cornell.edu

Abstract. From the emissions of massive quasars scattered across the universe, to the fluctuations in the stock market and the melodies of music, several real world signals have a power spectral density (PSD) that follows an inverse relationship with their frequency. Specifically, this type of random process is referred to as a $1/f$ signal, and has been of much interest in research, as sequences that have this property better mimic natural signals. In the context of constraint programming, a recent work has defined a constraint that enforces sequences to exhibit a $1/f$ PSD, as well as a corresponding constraint propagator. In this paper we show that the set of valid solutions associated with this propagator misses an exponential number of $1/f$ solutions and accepts solutions that do not have a $1/f$ PSD. Additionally, we address these two issues by proposing two non-exclusive algorithms for this constraint. The first one can find a larger set of valid solutions, while the second prevents most non-$1/f$ solutions. We demonstrate in our experimental section that using the hybrid of these two methods results in a more robust propagator for this constraint.

1 Introduction

The power spectral density (PSD) of a signal describes the distribution of power over its consisting frequency components, and is an extremely useful metric for the analysis of stochastic processes. Interestingly, the PSD of many real world phenomena, including quasar emissions, the firing of neurons and the resistivity of semiconductors, exhibit a signal that gives a PSD that is inversely proportional to its frequency [7]. To be precise this means that the PSD value follows a $1/f$ shape, where f is the frequency, and is called $1/f$ noise accordingly. The synthetic generation of $1/f$ signals can be of much use in content generation, for example it can help improve the digital generation of images [14]. Another example, and one of the most interesting phenomena observed, is that music that follows a PSD of $1/f$ empirically sounds better than music that has other distributions [23,24]. Specifically, music generated using $1/f$ sequences sounds less artificial [5]. Given the potential benefits of generating $1/f$ noise, several methods have been proposed [6,7]. A notably interesting algorithm is the Voss algorithm, which was developed by the physicist Richard Voss and then later published by Gardner [4]. While this method can generate sequences that have

J. Hooker (Ed.): CP 2018, LNCS 11008, pp. 601–610, 2018.
https://doi.org/10.1007/978-3-319-98334-9_39

the $1/f$ property, it only captures a subset of possible solutions [1]. Furthermore, some of the generated sequences are not guaranteed to be $1/f$. One particularly interesting attempt in CP to embed the $1/f$ property into generated music uses the Voss algorithm to produce the Voss constraint [9]. This is interesting as while there are many works in CP that develop efficient models for generating music and text [3,9–11,18,21,22], embedding the constraint of generating $1/f$ noise in constraint programming solvers has been less explored. Specifically, the Voss constraint is a hard constraint defined over a list of variables that enforces that the assignment of the variables follows $1/f$ noise and utilizes the Voss algorithm. It therefore inherits the limitations of the Voss algorithm, namely the fact that it misses several valid sequences and can generate invalid solutions. The first issue of missing valid sequences is a problem that results from using the same initial synchronization for all sequences, and of hard encoding the frequency of change in the sequence. The second issue of generating invalid solutions is a typical problem of probabilistic modeling in CP.

Out Contributions: (1) We propose to tackle the synchronization problem by allowing a "shifting effect" in our constraint and can thus generate an exponentially larger set of $1/f$ sequences [1]. **(2)** Additionally, we address the frequency problem by modeling the problem in a completely probabilistic way. We note that the use of the statistical properties to enforce constraints has been demonstrated in many works [8,10,12,13,16,17,19,20]. **(3)** We address the second issue of not generating invalid solutions by proposing a hybrid method of our two models. **(4)** We demonstrate the better performance of our two methods and hybrid method experimentally, by generated sequences.

2 Preliminaries

$1/f$ ***Spectrum*** Fourier Analysis is an extremely useful analysis tool that decomposes a signal into its frequency components. In particular, one powerful metric that uses Fourier Analysis, called power spectral density (PSD), analyzes how the power of a signal is distributed over its frequency components. Typically, the metric is displayed graphically with the power of the signal plotted against frequency. The PSD of a signal can give intuition into how the signal behaves, and if it is primarily made up of low or high frequencies. For example, a signal with a flat PSD, meaning an even power distribution across all frequencies, represents an entirely random process. A real world example of this kind of signal would be white noise, where each successive value in the sequence is picked in a random way. On the opposite end of the spectrum is Brownian Motion, which is a process where each successive value in a sequence is assigned by a small random fluctuations in the previous value. The result is that Brownian Motion is primarily described by lower frequencies and thus the PSD of such a signal follows the curve of $\frac{1}{f^2}$, where f is frequency. Using the same style of description, i.e. $\frac{1}{f^\alpha}$, white noise is simply the case when $\alpha = 0$ and Brownian noise is described when $\alpha = 2$. Thus $1/f$ spectrum is just the case in this description where $\alpha = 1$, and is sometimes called pink noise. Intuitively, it describes a signal

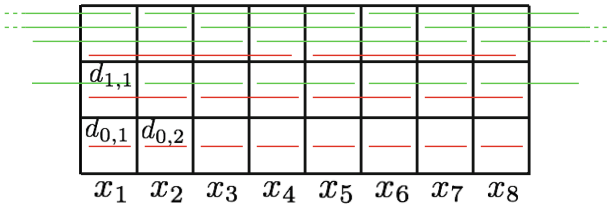

Fig. 1. Each segment represents a rolled die value. In red, no shift (previous model). In green, all possible shifts (our model). (Color figure online)

that is a hybrid of white noise and Brownian motion, in that it is mostly made up of low frequencies, but still has some high frequency components. As discussed before $1/f$ noise is an interesting special case because many natural signals have PSD's that follow the $1/f$ curve. For this reason it is highly valuable to have an algorithm capable of generating $1/f$ sequences.

Voss Algorithm. The Voss algorithm generates $1/f$ sequences by modeling the problem using dice. Essentially, to generate a sequence of length n it uses n iterations of rolling $\approx \log_2(n)$ dice, where each die is re-rolled every 2^i turns and i is the number of the die. At each iteration, the value assigned to the corresponding variable in the sequence is given by the summation of the dice values. To better illustrate the algorithm we unroll it in the table on the right of this paragraph to generate a $1/f$ sequence of length 8, using 3 dice referred to as the red (R), green (G) and blue (B) die as described by Gardner [4].

In the first iteration the R, G and B dice are rolled and have values of 3, 2 and 5 respectively. The first value in the sequence is just the summation of the three values and is thus ten. Additionally, the Binary representation of the iteration number is shown, where each bit corresponds to a die. Since we are starting at iteration 0, all the bits are set to 0. For each successive iteration, the binary representation is incremented by 1, to reflect the new iteration value. Then the dice are re-rolled if their respective bit in this representation flips value, resulting in one, two or all three dice changing value. Lastly, the new dice values are summed and the resulting value is assigned to the next variable in the sequence. This algorithm provides a framework for generating $1/f$ sequences by assigning each successive variable each iteration, but in order to be used in CP it needs to be adapted into a constraint.

Iter	Binary	Dice Roll	Val
i	R G B	R G B	v
0	0 0 0	3 2 5	10
1	0 0 **1**	3 2 **4**	9
2	0 **1 0**	3 **1 2**	6
3	0 1 **1**	3 1 **6**	10
4	**1 0 0**	**6 5 3**	14
5	1 0 **1**	6 5 **4**	15
6	1 **1 0**	6 **1 2**	9
7	1 1 **1**	6 1 **5**	12

Voss Constraint. The Voss constraint has been introduced in [9] and is named after the physicist Richard Voss, because its implementation uses the Voss algorithm. It is defined in the following way, let $X = \{x_1, x_2, ..., x_n\}$ be a list of

variables, let K be the number of dice and R be the maximum value of the dice. Then the Voss constraint, applied to X, ensures that the values of the variables X could have been obtained by rolling dice following the Voss algorithm [4]. Red segments from Fig. 1 represent rolled dice values. As shown, the rolled die value $d_{0,1}$ is defined only for x_1, while the rolled die value $d_{1,1}$ is define for both x_1 and x_2. For a given variable, its value is defined by the sum of all the rolled dice values above. For example $x_1 = d_{0,1} + d_{1,1} + d_{2,1}$, while $x_5 = d_{0,5} + d_{1,3} + d_{2,2}$. Specifically, the constraint is defined by the following: let d be the list of dice and let $d_{i,j}$ be the j^{th} rolling value of the i^{th} die then solutions of the constraint ensure that variables in $x_i \in X$ follow the equation:

$$x_i = \sum_{j=0}^{K-1} d_{j,\lceil \frac{i}{2^j} \rceil} \tag{1}$$

The tree shaped structure emanating from the red segment is called the Voss tree, and it represents all the possible solutions of the propagator proposed by [9]. Their propagator consists in constructing the tree shaped ternary sum constraint network. While this method does generate $1/f$ sequences, it only captures a subset of possible solutions [1] as it does not consider different initial conditions or what this paper refers to as dice shifting.

3 Shifted Dice

As known and explicated in [1], the dice rolling configuration from [9] is not the only method to generate $1/f$ sequences. In the original method each die is re-rolled exactly every 2^i turns, where i is the number of the die. While this ensures the generation of $1/f$ noise, it fails to consider that the dice do not need to have the same count for the number of turns. For example die 2 and die 3 should be re-rolled every 2^2 and 2^3 turns respectively, but there is no reason they must have the same count of turns. By allowing for a different counting of turns for each die, this essentially allows for the dice to be shifted with respect to one another. In terms of the algorithm show in 2, this has the same effect as starting with a non zero iteration, i.e. non zero binary representation. In this way, the previous method is actually a special case of the shifted version, where all the dice are in phase with one another. Whereas, in this method we allow each die to be treated independently and we allow for all possible initial shifted conditions.

Green segments from Fig. 1 represent the missing shifts. From a CP point of view, this implies that the proposed definition and propagator are too restrictive and miss an exponential number of solutions. In this section, we aim to model this shifting behaviour and translate it into constraints. Given a vector $S = \{s_1, ..., s_K\}$ representing the shift of each die, the general formula value of a variable is given by:

$$x_i = \sum_{j=0}^{K-1} d_{j,\lceil \frac{i+s_j}{2^j} \rceil} \tag{2}$$

We used a modified formula derived from [9] as it allows us to have each die indexed by its frequency. First, the S values represent the current state of a die. It is important to note that values of s_j that are $\geq 2^j$ or are $s_j < 0$, are equivalent to the value $s_j \mod 2^j$. Let the following conditional variable C be defined by: $C = i + s_j < 2^j$. Then the equation for any variable becomes:

$$x_i = \sum_{j=0}^{K-1} C d_{j,\lceil \frac{i}{2^j} \rceil} + \overline{C} d_{j,1+\lceil \frac{i}{2^j} \rceil} \tag{3}$$

Proposition 1. *Domain propagation of the Voss constraint with shift is NP-Complete.*

Proof. Consider the subset sum problem, let $V = \{v_1, v_2, ..., v_k\}$ be the set possible values and a be the desired value for the sum. The subset sum problem consists in finding a subset $V' \in V$ such that $\sum_{v \in V'} v = a$. Consider the Voss constraint, applied to the variables list X, using the following fixed values for the dice, $d_{i,1} = v_i$ and $d_{i,2} = 0$ for all i in $[1, \mathrm{k}]$. Let $x_2 = a$, enforcing domain consistency implies solving the subset sum problem. The hint here is that for a given variable we will have to choose between the non-shifted dice or the next one, resulting in an exponential number of choices.

While this propagator can generate an exponentially larger set of solutions compared to the previous definition, it still fails to generate all the $\approx 1/f$ sequences. This is mainly because even this shifted version is a $1/f$ approximation algorithm, and the exact distance of 2^i for each die is restrictive. Moreover, by only defining the dice variable to be independent, this may end up in sequences that are not $1/f$, because we do not enforce the dice to actually change value. This means that it is possible, for example, that a dice i could have the same value for all variables, which would give a signal that does not have a $1/f$ PSD. To combat these issues we developed a probabilistic version of this model that can enforce the dice to change value.

4 Probabilistic Dice

The important part of the Voss algorithm is the frequency in which dice are re-rolled. In this section we translate the re-rolling frequency into a probability, and use the probability as a constraint. First, we define the notion of state, which represents the current value of the dice. A state, defined by S, is a list of value $S_i \in [a, b]$. Thus each S_i represents the value of a die. Then, the value of a variable x_j is given by: $x_j = \sum_{i=0}^{|S^j|} S_i^j$.

The frequency of re-rolling dice i is 2^i, thus the probability of modifying S_i, from one state to another is $P(S_i^j \neq S_i^{j+1}) = \frac{1}{2^i}$. This gives us the probability to change the value of each element of S. The new values are chosen using a uniform random distribution, $\forall v \in [a, b], P(v) = \frac{1}{b-a+1}$. Using the probability of changing a value and the probability for getting a new value, we define the state transition process.

Definition 1. *Given a state S^t, the next state S^{t+1} is defined as follow:*

$$\forall v \neq S_i^t, P(S_i^{t+1} = v) = P(S_i^t \neq S_i^{t+1}) * P(v) = \frac{1}{2^i(b-a+1)} \tag{4}$$

$$P(S_i^{t+1} = S_i^t) = 1 - \sum_{j=1}^{b-a} P(S_i^t \neq S_i^{t+1}) * P(j) = 1 - \frac{b-a}{2^i(b-a+1)} \tag{5}$$

Let M (Eqs. (4, 5)) be a Markov process, let $d_i, \forall i \in [1, P]$ be the a list of variables of size n representing the value of the ith die for each variable. We define a Markov constraint, using M and its automaton, by list d_i with lower and upper bounds on the product of their probability [12,13]. By using the bounding of the probability as a constraint we can control the number of times the dice are re-rolled. Using this definition, we can generate all possible solutions, in fact without any constraint on the bound, we can generate any sequence, even non-$1/f$ solutions.

The density of $1/f$ solutions becomes a problem with this method as the sequence length becomes large. By setting an upper bound and/or a lower bound on the probability, we can constrain the number of time a dice is rolled to a new value. However, we do not constrain, how the re-rolls are distributed throughout the sequence. Given a die i with probability $\frac{1}{2^i}$ to be rolled, we can accept $k = \lceil \frac{n}{2^i} \rceil$ re-rolling over a sequence of size n. The number of possible sequences where the die is rerolled k times is $\binom{n}{k}$. Consider that we are looking for the sequence where the space between two re-rolls is exactly 2^i.

Proposition 2. *Let the solution density be defined by $D(n,i) = \frac{2^i}{\binom{n}{k}}$. Then*

$$\lim_{n \to \infty} D(n,i) = 0$$

Proof. Since 2^i is fixed, we need to prove that $\binom{n}{k}$ grows to infinity as n grows. While k does depend on n, it can only have two values when n is incremented, k or $k+1$. Thus we need to prove both that $\binom{n}{k} < \binom{n+1}{k}$ and $\binom{n}{k} < \binom{n+1}{k+1}$. The first case is trivial. For the second case, we have

$$\binom{n}{k} \cdot \frac{n+1}{k+1} = \frac{n!}{k!(n-k)!} \cdot \frac{n+1}{k+1} = \frac{(n+1)!}{(k+1)!(n-k)!} = \binom{n+1}{k+1}$$

Finally, $k < n \implies \frac{n+1}{k+1} > 1 \implies \binom{n}{k} < \binom{n+1}{k+1}$.

Thus with longer sequences, it becomes more likely that this method will find a non-$1/f$ solution. Such an issue can be solved by using higher order Markov processes and manually remove some state/transition, which is close to define a regular constraint on the time-series constraint [2] on die using the regular expression $= {}^{[0,2^i]} \neq (={}^{2^i \pm \epsilon} \neq)^*$. For example, if $\epsilon = 1$ the generated sequences have distances of 2^{i-1}, 2^i, or 2^{i+1} between reroll. Given that the shifted version of our propagator cannot constrain that the dice actually change value, and

that the probabilistic model cannot constrain the distributions of dice re-rolls, it seemed natural that a hybrid method of the two would give the best results.

The Hybrid Method. In order to solve the issue of solution density in the probabilistic model, and the issue of non-$1/f$ solutions produced by the dice shifting model, we propose to combine the two methods into a single hybrid method. The hybrid method we propose uses the model defined for the shifted dice, but in addition, applies the Markov constraint to the shifting dice. In this way the Markov constraint will be used to give a lower bound to the probability that the dice change value in the shifted dice framework. Specifically we know that the probability to change a dice value is $p_c = \frac{b-a}{b-a+1}$, and to keep the same value is $p_k = \frac{1}{b-a+1}$. One of the advantages to this method is that there are less variables in the d_i lists from the shifted version than the probabilistic model. However, one drawback of this method is that we are enforcing there to be exactly 2^i turns between re-rolling, which is restrictive.

5 Sequence Generation

In the same manner as [9], we sample $1/f$ sequences using a CP solver and the newly introduced propagators. Our goal is to show the following points: **(1)** The shifted version of our propagator generates $1/f$ sequences, while being able to generate a larger scope of solution than its predecessor. **(2)** The probabilistic version, while working well outside of CP, fails to generate $1/f$ sequences. **(3)** Our hybrid approach is able to generate $1/f$ sequences, and is more robust than the simple shifted version.

Protocol. We aim at generating sequences of length 10,000 using $13 \approx \log_2(10,000)$ dice. We use Google ORtools solver [15], and we use arithmetic constraints for the shifted model and the Markov constraint [13] for the probabilistic model. The search is a fully random one.

Results. We tested our three methods, the shifted dice, the probabilistic formulation and the hybrid method, both in CP solvers and in a stand alone process. Specifically we want to compare how the fixed frequency methods, i.e. the shifted dice, and hybrid method, compare to the probabilistic method. As seen in Fig. 3 we show the PSD of the original algorithm, the shifted dice version and the hybrid method, all plotted in log-log scale, where a line with slope -1 indicates $1/f$ noise. As can be seen in the figure the shifted dice method successfully produces $1/f$ noise both using a CP solver and as a standalone process. Additionally, the hybrid method successfully produces $1/f$ noise using a CP solver. This result is expected as the shifted dice method is simply a generalization of the original method which was already shown to produce $1/f$ sequences. With these results we believe we have shown that our general shifted dice method, and our hybrid method successfully generate $1/f$. Further, because of their theoretical formulation we know that they are capable of capturing an exponentially larger set of viable $1/f$ solutions. The time for generating a 10,000 long

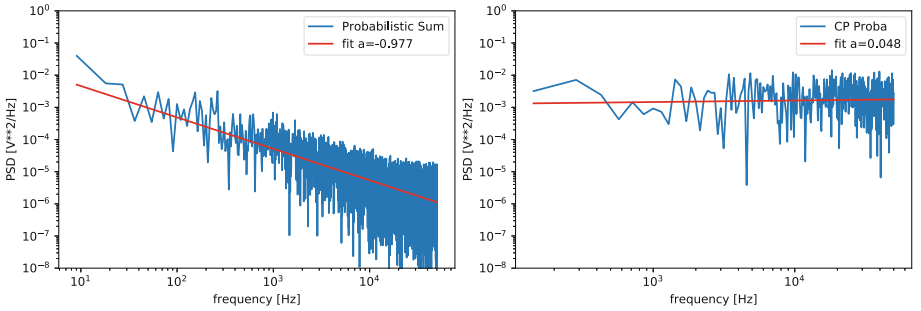

Fig. 2. PSD of probabilistic method in Left: standalone generation Right: in CP solver.

sequences is less than 2 s. For this sample the shift of each dice is given by $[0, 0, 3, 4, 9, 23, 41, 13, 182, 198, 263, 1794, 949]$.

Figure 2 shows the PSD of signals produced using only the probabilistic method, both within a CP solver and as a standalone process. As can be seen as a standalone process the probabilistic method is successful at generating $1/f$ sequences. However, when formulated for use in a CP solver, this approach fails to generate $1/f$ noise. In fact, it appears to generate white noise, given the small slope. Upon closer inspection of our results, it seems the random search very

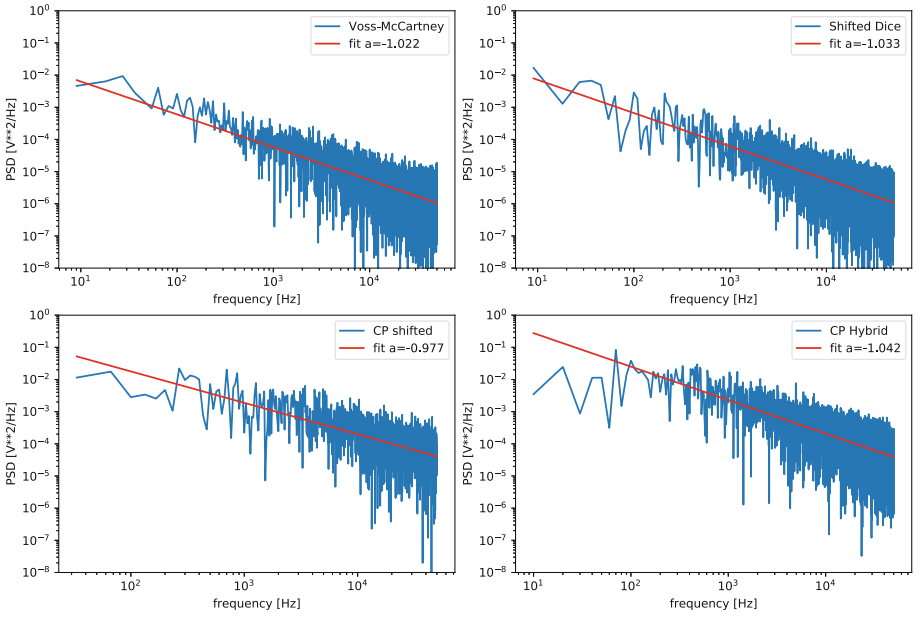

Fig. 3. Top left: PSD of Voss algorithm. Top right: PSD of shifted dice method outside of CP solver. Bottom left: PSD of shifted dice method using CP solver. Bottom Right: PSD of hybrid method using CP solver.

quickly modifies the dice values, and within a relatively small number of assignments reaches its maximum number of modifications as given per the probability constraint. Thus such a constraint alone is not enough for solving the $1/f$ problem. Additionally, because of the larger number of variables (#dice * sequence length) the time required for generating is substantially larger, but remains less than 2 min.

We believe, given these results, that the hybrid model is the most promising for generating $1/f$ sequences. This method uses the shifted model with the additional Markov constraints, which enforce that the dice change values. We found that this model is as fast as just the shifted method and we believe that it ensures a certain robustness for generating $1/f$ sequences, as well as preventing outlier solutions. Specifically it removes the ability of the shifted version to produce non-$1/f$ solutions. Moreover, the Markov constraint on the dice is less restrictive than a GCC, enforcing to know in advance the values.

6 Conclusion

In this paper we extend the capacity of the Voss constraint definition and give two different propagators, as well as a hybrid of the two, for enforcing it. Specifically, we show how to generalize a Voss tree, to allow each shifted version. Additionally, we answer an open question from [9] by proposing an entirely probabilistic CP model for generating $1/f$ sequences. We experimentally show that this approach fails to generate $1/f$ sequences in a CP solver, as the solution density is very low. However, this method does work outside of CP solvers. Finally, we define a hybrid method combining the strengths of both propagators, and show its robustness for generating $1/f$ sequences.

References

1. Herriman, A., McCartney, J., Burk, P., Downey, A., Whittle, R., Kellet, P.: Generation of pink (1/f) noise. http://www.firstpr.com.au/dsp/pink-noise/
2. Arafailova, E., et al.: Global constraint catalog, vol. II, time-series constraints. arXiv preprint arXiv:1609.08925 (2016)
3. Chemillier, M., Truchet, C.: Computation of words satisfying the rhythmic oddity property (after simha arom's works). Inf. Process. Lett. **86**(5), 255–261 (2003)
4. Gardner, M.: White and brown music, fractal curves and one-over-f fluctuations. Sci. Am. **238**(4), 16–32 (1978)
5. Hennig, H., et al.: The nature and perception of fluctuations in human musical rhythms. PLoS ONE **6**(10), e26457 (2011)
6. Kasdin, N.J.: Discrete simulation of colored noise and stochastic processes and 1/f/sup/spl alpha//power law noise generation. Proc. IEEE **83**(5), 802–827 (1995)
7. Keshner, M.S.: 1/f noise. Proc. IEEE **70**(3), 212–218 (1982)
8. Morin, M., Quimper, C.-G.: The markov transition constraint. In: Simonis, H. (ed.) CPAIOR 2014. LNCS, vol. 8451, pp. 405–421. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07046-9_29

9. Pachet, F., Roy, P., Papadopoulos, A., Sakellariou, J.: Generating 1/f noise sequences as constraint satisfaction: the voss constraint. In: IJCAI, pp. 2482–2488 (2015)
10. Papadopoulos, A., Pachet, F., Roy, P., Sakellariou, J.: Exact sampling for regular and markov constraints with belief propagation. In: Pesant, G. (ed.) CP 2015. LNCS, vol. 9255, pp. 341–350. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23219-5_24
11. Perez, G., Régin, J.-C.: Efficient operations on MDDs for building constraint programming models. In: IJCAI, pp. 374–380 (2015)
12. Perez, G., Régin, J.-C.: MDDs are efficient modeling tools: an application to some statistical constraints. In: Salvagnin, D., Lombardi, M. (eds.) CPAIOR 2017. LNCS, vol. 10335, pp. 30–40. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59776-8_3
13. Perez, G., Régin, J.-C.: MDDs: sampling and probability constraints. In: Beck, J.C. (ed.) CP 2017. LNCS, vol. 10416, pp. 226–242. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66158-2_15
14. Perlin, K.: An image synthesizer. ACM Siggraph Comput. Graph. **19**(3), 287–296 (1985)
15. Perron, L.: Or-tools. In: Workshop CP Solvers: Modeling, Applications, Integration, and Standardization (2013)
16. Pesant, G.: Achieving domain consistency and counting solutions for dispersion constraints. INFORMS J. Comput. **27**(4), 690–703 (2015)
17. Rossi, R., Prestwich, S., Tarim, S.A.: Statistical constraints. arXiv preprint arXiv:1402.5161 (2014)
18. Roy, P., Pachet, F.: Enforcing meter in finite-length markov sequences. In: AAAI (2013)
19. Schaus, P., Deville, Y., Dupont, P., Régin, J.-C.: The deviation constraint. In: Van Hentenryck, P., Wolsey, L. (eds.) CPAIOR 2007. LNCS, vol. 4510, pp. 260–274. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72397-4_19
20. Schaus, P., Régin, J.-C.: Bound-consistent spread constraint. EURO J. Comput. Optim. (2013)
21. Truchet, C., Assayag, G.: Constraint Programming in Music. ISTE-Wiley, Hoboken (2011)
22. Truchet, C., Codognet, P.: Musical constraint satisfaction problems solved with adaptive search. Soft. Comput. **8**(9), 633–640 (2004)
23. Voss, R.F., Clarke, J.: 1/f noise in music and speech. Nature **258**(5533), 317–318 (1975)
24. Voss, R.F., Clarke, J.: 1/f noisein music: Music from 1/f noise. J. Acous. Soc. Am. **63**(1), 258–263 (1978)